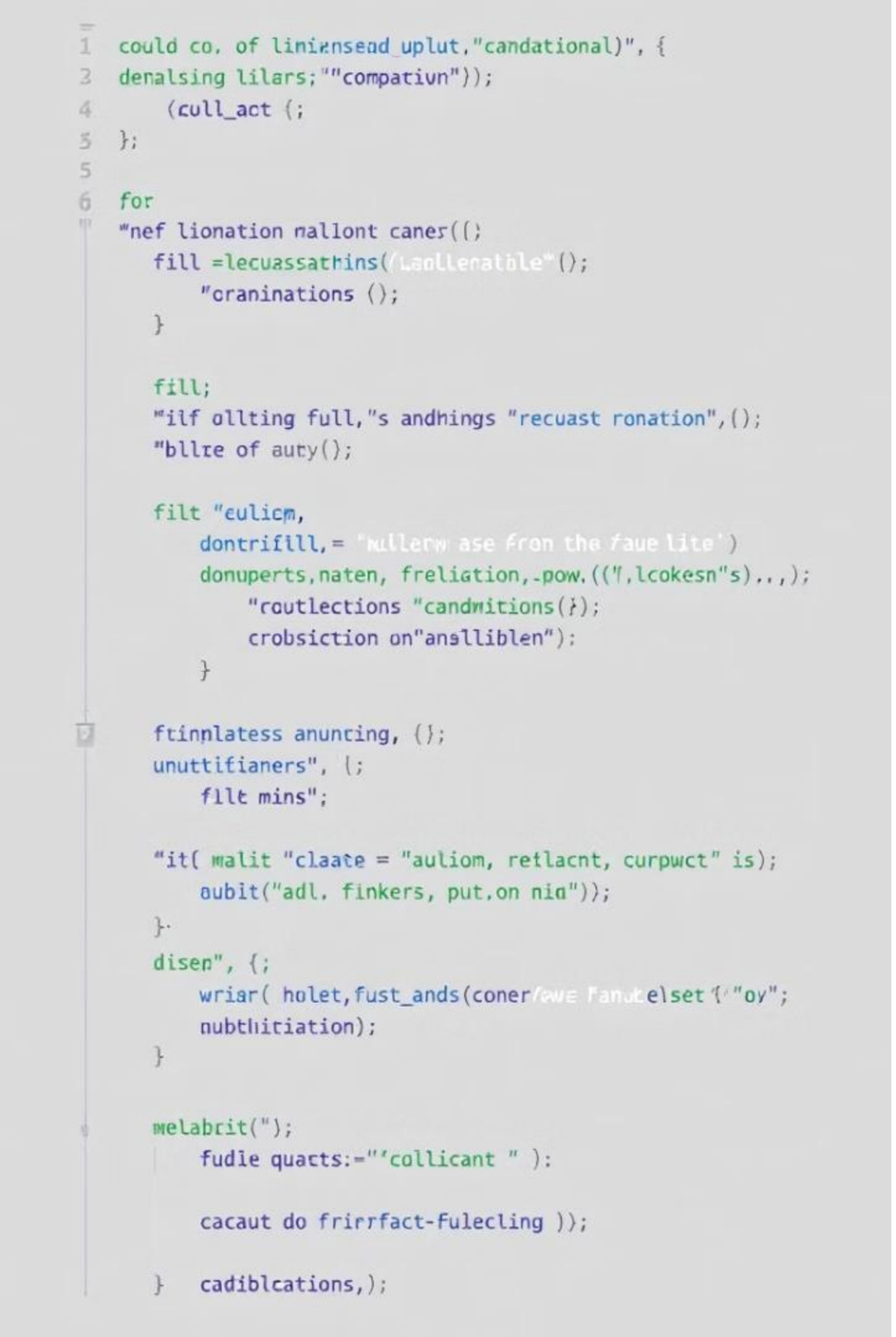
Методи рефакторингу коду програмного забезпечення

Вітаю! Сьогодні розглянемо три методи рефакторингу коду, що застосовуються для покращення якості програмного забезпечення: Extract Method, Rename Method та Move Method. Ми ознайомимося з їхнім призначенням, прикладами використання, а також обговоримо переваги, які вони надають для підтримуваності та розширюваності коду.

Шеремет Андрій Григорович
ПЗПІ-22-6

05.01.2025



Вступ

Мета роботи:

Навчити студентів основним методам рефакторингу коду на основі реальних прикладів з їхніх власних програмних проєктів. Студенти повинні навчитися ідентифікувати проблеми в коді та використовувати відповідні методи рефакторингу для покращення його якості.

Обрані методи:

Extract Method (Виділення методу)

Rename Method (Перейменування методу)

Move Method (Переміщення методу)

Extract Method (Виділення методу)

Проблема:

У початковій версії коду метод може бути занадто довгим або виконувати кілька різних завдань, що знижує його читабельність, ускладнює підтримку та підвищує ймовірність помилок. Довгі методи важко зрозуміти, оскільки вони містять багато логіки в одному місці, що змушує програмістів витрачати більше часу на аналіз коду. Крім того, модифікація або повторне використання частини довгого методу в іншому місці вимагає копіювання коду, що призводить до його дублювання та ускладнює подальші зміни. Метод Extract Method є оптимальним у таких випадках, оскільки дозволяє виділити окремі логічні частини великого методу в невеликі методи з чіткими іменами. Це покращує читабельність коду, зменшує дублювання та спрощує його підтримку й розширення.

Код до рефакторингу:

```php
function generateReport($orders) {
    $totalAmount = 0;
    foreach ($orders as $order) {
        echo "Order ID: " . $order['id'] . "\n";
        echo "Customer: " . $order['customer_name'] . "\n";
        echo "Amount: " . $order['amount'] . "\n";
        $totalAmount += $order['amount'];
    }
    echo "Total Amount: " . $totalAmount . "\n";
}
```

Extract Method (Виділення методу)

Пояснення методу:

Метод Extract Method дозволяє покращити структуру коду шляхом винесення окремих логічних частин із великого методу в невеликі, зрозумілі методи з чіткими завданнями. У початковій версії код був перевантажений різними завданнями: виведення інформації про замовлення та обчислення загальної суми виконувались в одному місці, що ускладнювало його читання, підтримку та повторне використання. Виділення окремих методів усунуло ці проблеми, зробивши код більш читабельним, зрозумілим і зручним для модифікації. Тепер кожна функція відповідає лише за одне завдання, що спрощує тестування й дозволяє уникнути дублювання коду у майбутньому.

Код після рефакторингу:

```php
function generateReport($orders) {
    printOrders($orders);
    $totalAmount = calculateTotalAmount($orders);
    echo "Total Amount: " . $totalAmount . "\n";
}

function printOrders($orders) {
    foreach ($orders as $order) {
        echo "Order ID: " . $order['id'] . "\n";
        echo "Customer: " . $order['customer_name'] . "\n";
        echo "Amount: " . $order['amount'] . "\n";
    }
}

function calculateTotalAmount($orders) {
    $total = 0;
    foreach ($orders as $order) {
        $total += $order['amount'];
    }
    return $total;
}
```

Extract Method (Виділення методу)

Переваги отриманого коду після застосування методу:

Після застосування методу Extract Method код став більш читабельним, оскільки кожна логічна частина винесена в окремий метод із чітким чітким призначенням. Це спростило розуміння функціоналу, поліпшило підтримуваність і розширюваність, оскільки будь-які зміни тепер можна вносити лише в окремі методи без ризику порушити роботу всього коду. Завдяки виділенню методів зменшилось дублювання коду, що дозволяє повторно використовувати окремі частини, а також покращилась структура програми, підвищуючи її модульність і спрощуючи тестування.

Rename Method (Перейменування методу)

Проблема:

Існуючий метод має неінформативну або некоректну назву, що ускладнює розуміння його призначення. Це може привести до помилок у використанні та ускладнити підтримку коду, оскільки інші розробники або навіть сам автор не можуть швидко зрозуміти, що саме робить метод, виходячи лише з його назви.

Код до рефакторингу:

```php
function update($data) {
    // Оновлює користувача в базі даних
    $user = getUserById($data['id']);
    if ($user) {
        $user->name = $data['name'];
        $user->email = $data['email'];
        saveUser($user);
    }
}
```

Rename Method (Перейменування методу)

Пояснення методу:

Метод "Rename Method" полягає в зміні назви методу, щоб зробити його його більш описовим і зрозумілим для інших розробників. Важливо, щоб назва методу чітко відображала його призначення і дії, які він виконує. Це підвищує читабельність та підтримуваність коду, адже інші розробники одразу розуміють, що саме робить метод, без необхідності аналізувати його реалізацію. Наприклад, у випадку з методом update, який оновлює дані користувача, можна змінити його назву на більш точну, наприклад, updateUserDetails.

Код після рефакторингу:

```php
function updateUserDetails($data) {
    // Оновлює інформацію про користувача в базі даних
    $user = getUserById($data['id']);
    if ($user) {
        $user->name = $data['name'];
        $user->email = $data['email'];
        saveUser($user);
    }
}
```

Rename Method (Перейменування методу)

Переваги отриманого коду після застосування методу:

Переваги отриманого коду після застосування методу Rename Method полягають у підвищенні читабельності та зрозумілості коду. Зміна назви методу на updateUserDetails робить його більш описовим і чітким, що дозволяє іншим розробникам або майбутнім командам швидше швидше зрозуміти, що саме робить цей метод без необхідності заглиблюватися в його реалізацію. Це зменшує ймовірність помилок, покращує підтримку та дозволяє легше масштабувати код, адже він стає більш зрозумілим і послідовним. Крім того, чітко названі методи полегшують тестування і рефакторинг, оскільки їх призначення стає очевидним.

Move Method (Переміщення методу)

Проблема:

Метод розташований у класі, де його функціональність не зовсім відповідає основному призначенню цього класу. Це може ускладнити підтримку коду і його розширення в майбутньому, оскільки метод займає місце в класі, де він не має прямого зв'язку з його основною логікою. Внаслідок цього інші методи або класи можуть залежати від методів, що не є їх частиною, що погіршує структуру програми.

Код до рефакторингу:

```php
class User {
    public $id;
    public $name;
    public $email;

    public function __construct($id, $name, $email) {
        $this->id = $id;
        $this->name = $name;
        $this->email = $email;
    }

    public function saveUser() {
        // Логіка збереження користувача в базі даних
        // ...
    }
}

class Order {
    public $id;
    public $userId;
    public $total;

    public function __construct($id, $userId, $total) {
        $this->id = $id;
        $this->userId = $userId;
        $this->total = $total;
    }

    public function getUserDetails() {
        $user = new User($this->userId, 'John Doe', 'john.doe@example.com');
        return $user;
    }
}
```

Move Method (Переміщення методу)

Пояснення методу:

Метод "Move Method" полягає в переміщенні методу з одного класу в інший, коли його функціональність більше підходить іншому класу. Це дозволяє покращити структуру програми та забезпечити дотримання принципу єдиної відповідальності, оскільки кожен клас відповідає лише за свою специфічну логіку. Переміщення методу в клас, де його логіка більш доречна, допомагає зменшити залежності між компонентами програми, покращити підтримку та зрозумілість коду.

Код після рефакторингу:

```php
class User {
    public $id;
    public $name;
    public $email;

    public function __construct($id, $name, $email) {
        $this->id = $id;
        $this->name = $name;
        $this->email = $email;
    }

    public function saveUser() {
        // Логіка збереження користувача в базі даних
        // ...
    }

    public static function getUserById($id) {
        // Логіка отримання користувача за ID
        return new User($id, 'John Doe', 'john.doe@example.com');
    }
}

class Order {
    public $id;
    public $userId;
    public $total;

    public function __construct($id, $userId, $total) {
        $this->id = $id;
        $this->userId = $userId;
        $this->total = $total;
    }

    public function getUserDetails() {
        $user = User::getUserById($this->userId);
        return $user;
    }
}
```

Move Method (Переміщення методу)

Переваги отриманого коду після застосування методу:

Переваги отриманого коду після застосування методу Move Method полягають у значному покращенні структури та організації програми. Переміщення методу в клас, який безпосередньо відповідає за його функціональність, дозволяє дотримуватися принципу єдиної відповідальності, зменшуючи складність класів і підвищуючи їх зрозумілість. Клас Order більше не містить логіки, яка не має відношення до його основної мети, що робить його більш спеціалізованим і легким для підтримки. Переміщення методу також зменшує кількість залежностей між класами, робить код більш модульним, що полегшує тестування та рефакторинг у майбутньому.

Висновки

Рефакторинг значно покращив якість коду, підвищивши його читаємість, підтримуваність і модульність. Зміна назв методів і їх переміщення до відповідних класів дозволила зменшити складність і покращити структуру програми. Кожен клас тепер виконує лише одну конкретну задачу, що відповідає принципу єдиної відповідальності. Це сприяє зменшенню ймовірності помилок, полегшує тестування та підтримку коду, а також робить його більш зрозумілим для інших розробників. Рефакторинг також дозволяє легше масштабувати програму, знижуючи рівень залежностей між класами і підвищуючи гнучкість у роботі з кодом.